SKIRANGER FÜR KINECT

Zwei Studenten der HTW Dresden entwickeln ein Ski-Racing-Spiel und die dazugehörige Bewegungssteuerung gleich mit. Im Artikel erzählen sie, wie sie Motion Capturing, das Unreal DevKit und Kinect unter einen Hut gebracht haben.

Toni Seifert
ist Mitgründer von Outpost Studios.

Toni widmete sich bei SkiRanger vorrangig dem Design und der Programmierung der Spielmechanik.

Henrik Weirauch
ist ebenfalls Mitgründer von Outpost Studios.

Henrik war hauptsächlich für die Konzeption und Programmierung der von SkiRanger unterstützten Trackingsysteme zuständig.

Wir sind beide Medieninformatikabsolventen der HTW Dresden und Entwickler des PC-Kinect-Titels »SkiRanger«. Ursprünglich entwarfen wir das Skispiel für das Motion-Capture-System unserer Hochschule und begannen bereits 2011 im Rahmen unserer Diplomarbeit mit der Umsetzung. Im Artikel berichten wir von einigen Herausforderungen, die wir bei der Realisierung des bewegungsgesteuerten Spiels meistern mussten.

Die Diplomarbeit

2009 bekam die HTW Dresden ihr eigenes Motion-Capture-System (MoCap), mit dem man allerlei Forschung in den Bereichen der Bewegungsanalyse und Animation betreiben konnte. Prof. Dr. Markus Wacker, der Hauptverantwortliche des neuen Systems, war zu dieser Zeit noch auf der Suche nach Projekten aus dem Bereich der Computerspiele. Ein solches Spiel sollte die Leistungsfähigkeit des MoCap-Systems demonstrieren, beispielsweise am Tag der offenen Tür. Außerdem wollte man die jüngeren Besucher begeistern. Das System erlaubt markerloses Tracking einer Person und schien nur darauf zu warten, als Eingabegerät für ein Spiel eingesetzt zu werden. Weil wir beide gerade auf der Suche nach einem spannenden Thema für unsere Diplomarbeit waren und auch schon einige Erfahrung mit der Unreal Engine hatten, kam eins zum anderen.

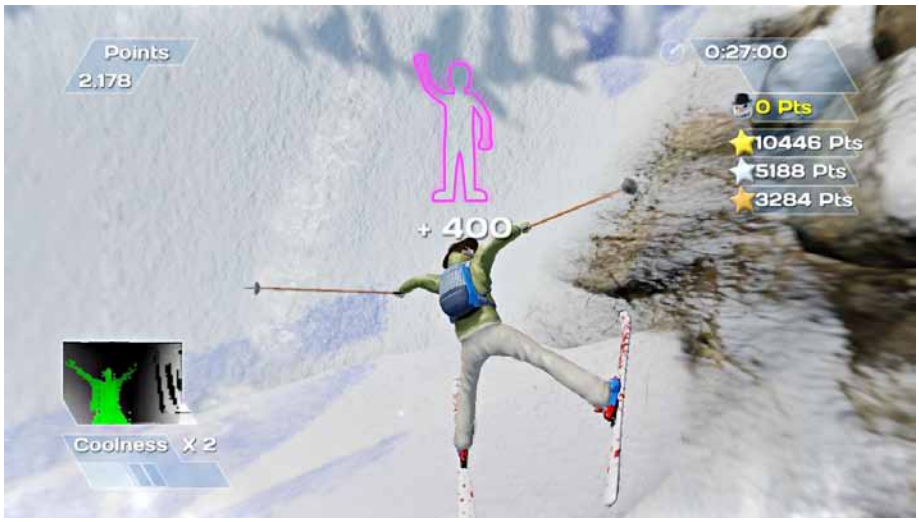
Unsere Aufgabe war es nun, ein Softwaremodul zu entwickeln, welches das »Stage« genannte Motion-Capture-System der Firma OrganicMotion und das Unreal Development Kit von Epic Games miteinander verband. Dabei sollte das Modul für andere Projekte der Hochschule wiederverwendbar sein, außerdem war die Entwicklung eines Spiels mit dieser Software-Kombination geplant. Dafür mussten wir zu Beginn herausfinden, welche Art von Spiel sich für solch eine Bewegungssteuerung eignen würde.

Das Motion-Capture-System

Das Besondere an dem verwendeten MoCap-System ist, dass es gänzlich ohne Marker auskommt, welche die Darsteller im Normalfall am Körper tragen müssen, um vom System erfasst zu werden. Stattdessen arbeitet das System mit 14 Infrarotkameras, die mit Hilfe des weißen retroreflektierenden Materials, das den Trackingbereich umgibt, die Silhouette des Benutzers aus unterschiedlichen Blickwinkeln erfasst und daraus dessen Bewegungen ableitet (**Abbildung 1**). Das System kann mit einer Framerate von 60, 90 oder 120 Hz arbeiten und besitzt eine Verzögerungszeit von etwa 50 bis 200 ms. Im Vergleich zu einer regulären Maus mit einer Verzögerungszeit von < 8 ms ist diese sehr lang und damit nur bedingt für Spieltypen geeignet, bei denen die Reaktionszeit des Spielers entscheidend ist. Erschwerend ist die Tatsache, dass dem Spieler kein klassisches Eingabegerät mit Knöpfen zur Verfügung steht, mit denen er ein zeitdiskretes Ereignis auslösen könnte, zum Beispiel um einen Schuss abzufeuern.

Das Unreal Development Kit

Die Werkzeuge des Unreal Development Kits erlauben es dem Entwickler unter anderem, externe Inhalte wie Texturen oder 3D-Modelle in die Engine zu importieren und sie für die weitere Nutzung im Spiel vorzubereiten. Außerdem ist es möglich, Teile des Unreal Editors mit Hilfe von UnrealScript zu erweitern. So hat man die Möglichkeit, für die integrierte visuelle Skriptsprache Kismet speziell an das eigene Projekt angepasste Nodes zu entwickeln, die es



SkiRanger ist nicht nur ein vollwertiges Racing Game inklusive drei Spielmodi, Multiplayer für vier Spieler und Trick Jumps, es lässt sich auch komplett ohne Gamepad mittels Körperbewegungen steuern.

dann den Leveldesignern auf einfache Weise ermöglichen, in den Spielverlauf einzugreifen. All das macht das UDK zu einem sehr mächtigen Werkzeug zu günstigen Konditionen, sofern man nicht den von Epic Games vorgesehenen Sandkasten verlassen möchte. Denn die einzige Schwäche, die wir für kleine Entwicklerteams ausmachen konnten, liegt in der mangelnden Unterstützung externer Frameworks.

In erster Linie benötigten wir eine Schnittstelle, um den Datenaustausch zwischen dem Motion-Capture-System und dem Spiel zu realisieren, damit wir die Daten als Eingabegerät und Animationsgrundlage nutzen konnten. Leider bietet das UDK in dieser Hinsicht nur wenig. Lediglich für die Zielplattform Windows gibt es die Möglichkeit, DLLs einzubinden, die über die Standard-C-Schnittstelle Funktionen exportieren. Diese DLLs müssen dann wiederum alle benötigten Bibliotheken kapseln. Die direkte Verwendung von C/C++, C#, Java oder anderen Sprachen ist mittels UDK nicht vorgesehen, weshalb es auch keine Frameworks oder Plugins gibt, die auf dem UDK aufsetzen.

Wir wollen Ski fahren

Warum es bei uns gerade ein Skispiel geworden ist, hängt damit zusammen, dass Skifahren einige Vorzüge für die Bewegungssteuerung mit sich brachte und uns manche Probleme elegant umgehen ließ. Da ist zum Beispiel die Frage der Fortbewegung: Wie wird die Spielfigur mit einer möglichst intuitiven Geste durch die Spielwelt bewegt, ohne sich vom Bildschirm entfernen zu müssen? Durch Laufbewegungen auf der Stelle? Skifahren umgeht das ganze Problem und die Skier selbst bieten der Spielfigur im wahrsten Sinne des Wortes eine Plattform, um die Piste hinunterzubrettern. Vereinfacht gesprochen muss der Spieler nur auf der Stelle stehen und kann die Spielfigur mittels Beugung und Drehung des Oberkörpers steuern. Folglich werden dafür auch keine Knöpfe benötigt. Im Skisport sind Sprünge und Tricks recht üblich, was uns die Möglichkeit bietet, Posen und Gesten des Spielers für diese Zwecke auszuwerten. Weiterhin bietet das Skifahren Vorteile gegenüber anderen vergleichbaren Sportarten mit Boards. So ist es dem Spieler im Gegensatz zum Snowboardfah-

Foto: Peter Sebb

Abbildung 1: Das markerlose Motion-Capture-System der HTW Dresden.

Foto: Robert Albert

Abbildung 2: Henrik beim Skifahren. Die Skiausrüstung dient natürlich nur Demonstrationszwecken …

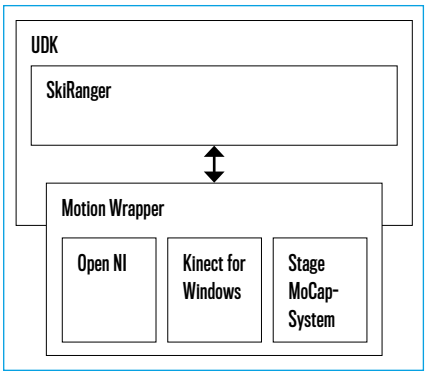


Abbildung 3: Die einzelnen Module bzw. die Architektur des SkiRanger-Projekts.

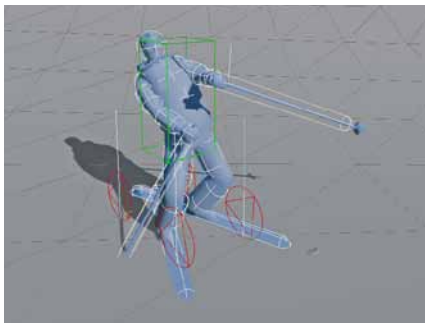


Abbildung 4: Das Skifahrzeug (grün) mit seinen unsichtbaren Rädern (rot) unter der animierten Spielfigur (weiß).

ren erlaubt, die Arme aktiv für die Fortbewegung einzusetzen (Abbildung 2).

Wo fängt man an?

Keine Frage, die Skifahrmechanik ist das zentrale Element des Spiels. Sie sollte intuitiv, fehlertolerant und zuverlässig mit den Bewegungsdaten arbeiten, sollte Raum für die Verbesserung der spielerischen Fähigkeiten bieten und natürlich Spaß machen. Doch das ist nur ein kleiner Teil des Projekts. Die anfangs wichtigste Aufgabe war die Verknüpfung von MoCap und UDK. Im Hinblick auf das Debugging war insbesondere die Arbeit mit der daraus resultierenden Schnittstelle selbst eine besondere Herausforderung. Wie man sich vorstellen kann, möchte man sich nicht jedes Mal in den Trackingbereich begeben, um eine bestimmte Funktionalität zu testen oder mit einer Geste ein bestimmtes Ereignis auszulösen. Deshalb mussten wir für diese Arbeit zusätzlich Werkzeuge entwickeln, um unser Spiel mit vorher aufgezeichneten Sequenzen testen zu können. Später entwickelten wir noch Systeme, um Posen des Spielers auszuwerten, die Menüs mit Gesten zu steuern, Bewegungsdaten und den Kalorienverbrauch aufzuzeichnen und danach grafisch auszuwerten. Nicht zu vergessen das Spiel selbst mit seinen Figuren und Levels (Abbildung 3).

Kommunikation von Trackingsystem und UDK

Zum Programmieren innerhalb des UDKs steht lediglich UnrealScript zur Verfügung. UnrealScript wird seit jeher in der Unreal Engine für die Realisierung der Spiellogik verwendet. Weil der Script-Code zur Laufzeit von der Unreal Virtual Machine interpretiert wird, ist er verglichen mit aus C++ generiertem Maschinencode verhältnismäßig langsam und damit für die Verarbeitung großer Datenmengen weniger gut geeignet. Das MoCap-System »bombardierte« uns aber regelrecht mit Knochenkoordinaten, die alle gepuffert, transformiert und interpoliert werden mussten. Möchte man gleichzeitig noch einen kleinen Videostream (von Kinect oder einer der 14 MoCap-Kameras) mit etwa 300.000 Farbwerten pro Frame bearbeiten, kommt man sofort an die Grenzen der Skriptsprache und die Framerate fällt ins Bodenlose. Deshalb blieb uns am Ende nichts anderes übrig, als auf die DLL-Bind-Funktionalität zurückzugreifen, um mit dem MoCap-System zu kommunizieren und zeitaufwändige Berechnungen in C++-Module auszulagern und teilweise in Windows-Threads zu parallelisieren. So entstand ein Softwaremodul, das die gesamte Kommunikation mit dem MoCap-System für den UDK-Entwickler unsichtbar macht und den Datenstrom für die weitere Verwendung im Spiel entsprechend aufbereitet.

Knochenkoordinaten für Spielsteuerung und Skelettanimation

In Computerspielen übernimmt der Spieler in der Regel nur bis zu einem gewissen Maß die Kontrolle über seine Spielfigur. Er bewegt bei einem Shooter beispielsweise nur einen Zylinder durch die Spielwelt. Die Spielfigur folgt diesem Zylinder und spielt die entsprechenden Animationen ab. Aufgrund der Charakteristik der Eingabegeräte ist es im Normalfall auch nicht sinnvoll, dem Spieler eine direkte Kontrolle über das rechte Bein seiner Spielfigur zu geben.

Bei SkiRanger ist das allerdings etwas anders. Das MoCap-System liefert weitgehend alle Informationen über die Haltung des Spielers, die man mehr oder weniger direkt auf die Spielfigur übertragen könnte, ohne dass der Spieler dabei umdenken müsste (z.B. muss man nicht Knopf X auf dem Gamepad drücken, um das rechte Bein zu bewegen). Trotzdem ist es auch hier nicht sinnvoll, aus diesen Bewegungsdaten die Bewegung der Spielfigur direkt abzuleiten oder zu übernehmen (ein Schritt in der Realwelt würde einen Schritt in der virtuellen Welt bedeuten).

Aus diesem Grund entschieden wir uns für eine Trennung von Steuerung der Skelettanimation der Figur und Steuerung der Spielfigur bezüglich der Skisimulation. Die Skisimulation übernimmt in unserem Fall ein unsichtbares »Skifahrzeug« unter der Spielfigur, bei dem die Fahrzeugsimulation des UDKs zum Einsatz kommt. Das Fahrzeug selbst wird in mehr oder weniger klassischer Weise gesteuert. Die Eingabeachsen für die Lenkung, Beschleunigung und das Bremsen werden dabei aus den Bewegungen des Spielers abgeleitet (Abbildung 4).

Für die Steuerung des Skifahrzeugs und die Animation des Skeletts mit Bewegungsdaten des Spielers mussten die Koordinaten zusätzlich in verschiedenen Bezugssystemen vorliegen. Das ist notwendig, um die Bewegungen von Spielern aller Größen auf Spielfiguren mit beliebigen Proportionen und Größen projizieren zu können (Abbildung 5). Außerdem musste ein flexibles lokales Koordinatensystem um den Spieler herum ermittelt werden, das seine aktuelle Position und Vorwärtsrichtung definiert. Denn das Spiel sollte innerhalb des Trackingbereichs nicht nur in eine Richtung gespielt werden können (das Bild sollte auch an die übrigen Wände projiziert werden).

Die Kinect-Portierung

Als wir unser Diplom endlich entgegennehmen konnten, hatten wir einen sehr gut funktionierenden Prototyp eines bewegungsgesteuerten Skispiels. Mit Ausnahme einer Präsentation auf der Siggraph 2011 (durch OrganicMotion) konnte SkiRanger jedoch nur auf einem Computer hinter verschlossenen Hochschultüren gespielt werden. So würden nur sehr wenige Menschen von unserer Arbeit Kenntnis nehmen. Deshalb wollten wir das Spiel einem breiteren Publikum zugänglich machen. 2011 war das Jahr der »Kinect-Welle« und wir schienen zur richtigen Zeit am richtigen Ort zu sein. Weil die Kinect große Ähnlichkeit mit dem MoCap-System bezüglich der bereitgestellten Datenströme besitzt, war die

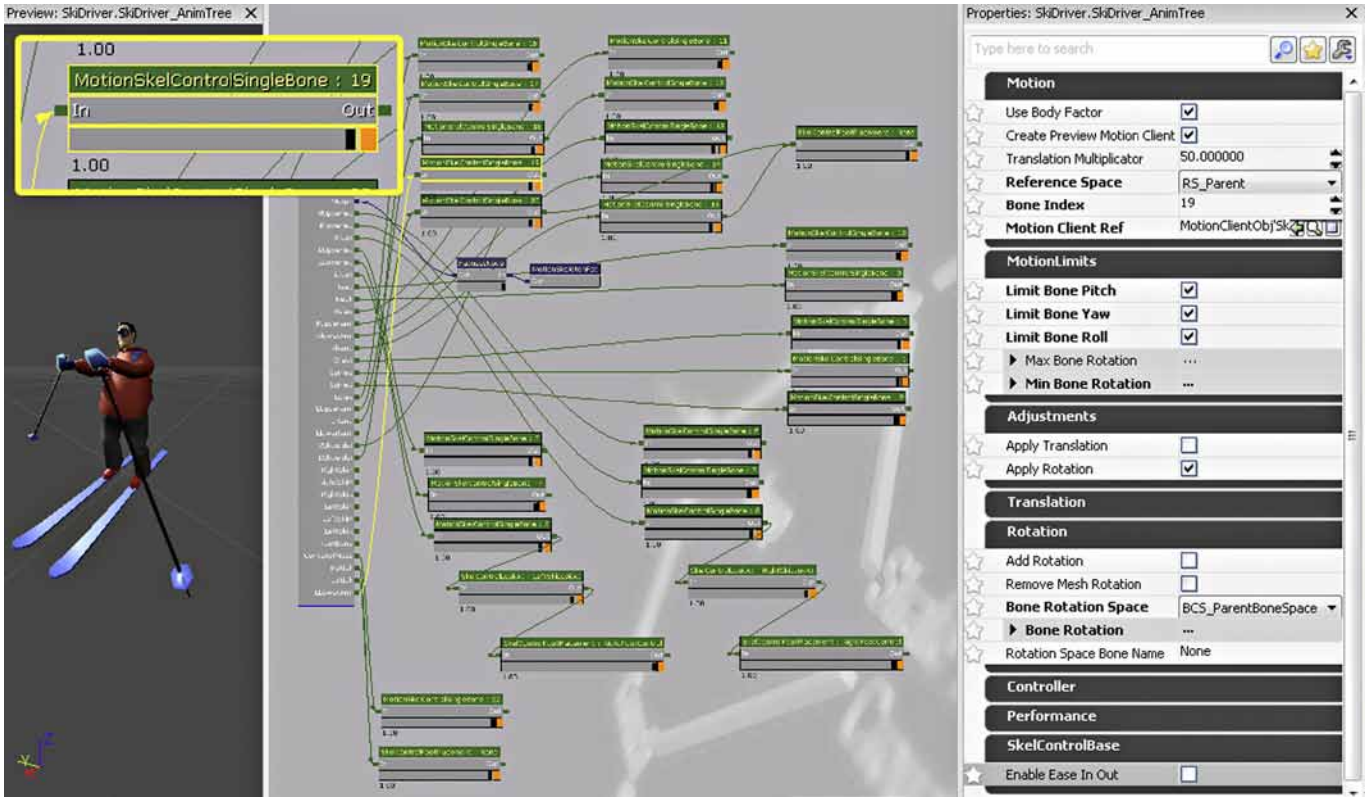


Abbildung 5: Der Animationsbaum im UnrealEditor. Mit speziell entwickelten Nodes können die Spielerbewegungen auf verschiedene Objekte übertragen werden.

Portierung aus technologischer Sicht der nächste logische Schritt für uns. Sobald die ersten Software-Frameworks für Kinect verfügbar waren, begannen wir mit der Arbeit. Unser Ziel war es, über einen typischen Kinect-Hack hinauszugehen und unser Spiel Schritt für Schritt zu einem vollwertigen Produkt zu entwickeln. Bei einer entsprechenden Entwicklung des Marktes wäre es sogar möglich, dass SkiRanger den Weg auf die Xbox findet.

Kameras und Frameworks

Die dynamische Entwicklung des Kinect-Markts in den Jahren 2010 bis 2012, sowohl im Bereich der Software-Frameworks als auch im Bereich Hardware, stellte uns vor die Herausforderung, in kürzester Zeit die wichtigsten Kamera-Frameworks zu implementieren.

Zu Beginn gab es viel Unklarheit darüber, ob die entwickelten Programme der Community für Kinect, sogenannte »Kinect Hacks«, legal waren und ob man diese kommerziell einsetzen dürfe, weil sie die Nutzung einer Xbox 360 Kinect voraussetzten. Um den kommerziellen Gebrauch zu ermöglichen, entwickelten PrimeSense und Microsoft zusätzlich eigene Sensorkameras, die jeweils nur mit ihren eigenen Frameworks funktionierten. Die beiden wichtigsten Vertreter dieser Frameworks sind somit OpenNI und das Kinect for Windows SDK. Statt eines großen Kinect-Sensor-Markts mit einheitlichen Software- und Hardwarestandards entstanden nun zwei kleine Märkte im PC-Bereich. Ein Markt für PC-Kinect (Kinect for Windows) und ein weiterer für ASUS Xtion (OpenNI). Der Einsatz der weit verbreiteten Kinect-Kamera bleibt Entwicklern kommerzieller Anwendungen für den PC weiterhin verwehrt.

Um den größtmöglichen Nutzerkreis zu erreichen, unterstützt SkiRanger mittlerweile jede erdenkliche Hardware-Software-Kombination im Rahmen der beiden Frameworks. Dank der MSSDK-OpenNI-Bridge von Tomoto Washio (Kinect Ultra Seven) ist sogar die Kombination von Kinect for Windows und OpenNI möglich, um die Vorteile beider Frameworks zu erhalten (siehe **Abbildung 6**).

Debugging der Bewegungssteuerung

Um die Bewegungsdaten der Kinect-Frameworks für unser Spiel nutzen zu können, mussten diese erst um einige Features erweitert werden. Während das MoCap-System die ungefähre Lage von 21 Knochen bereitstellt, liefert Kinect lediglich etwa 10 bis 14 Gelenkpunkte der getrackten Person, sofern diese sich im Sichtbereich der Kamera befinden. Können beispielsweise die

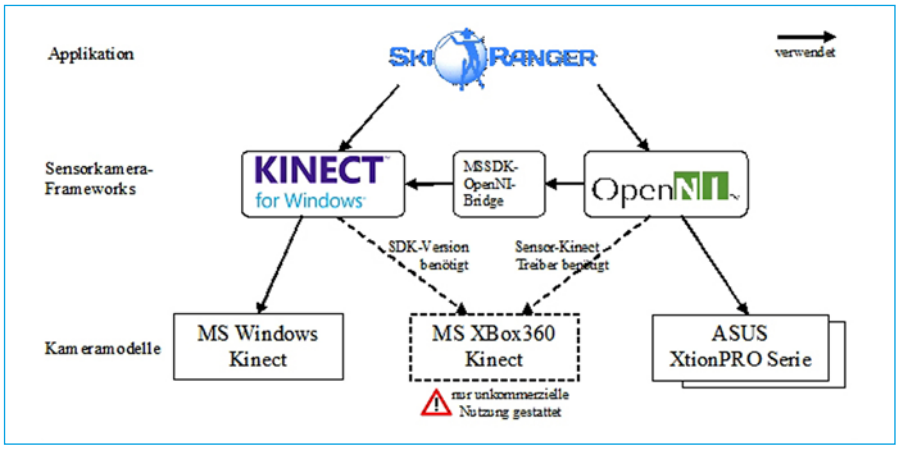


Abbildung 6: Von SkiRanger unterstützte Framework-Kamera-Kombinationen. Je nach vorgefundener Installation ermittelt das Spiel die beste Methode, mit der Kamera zu kommunizieren.

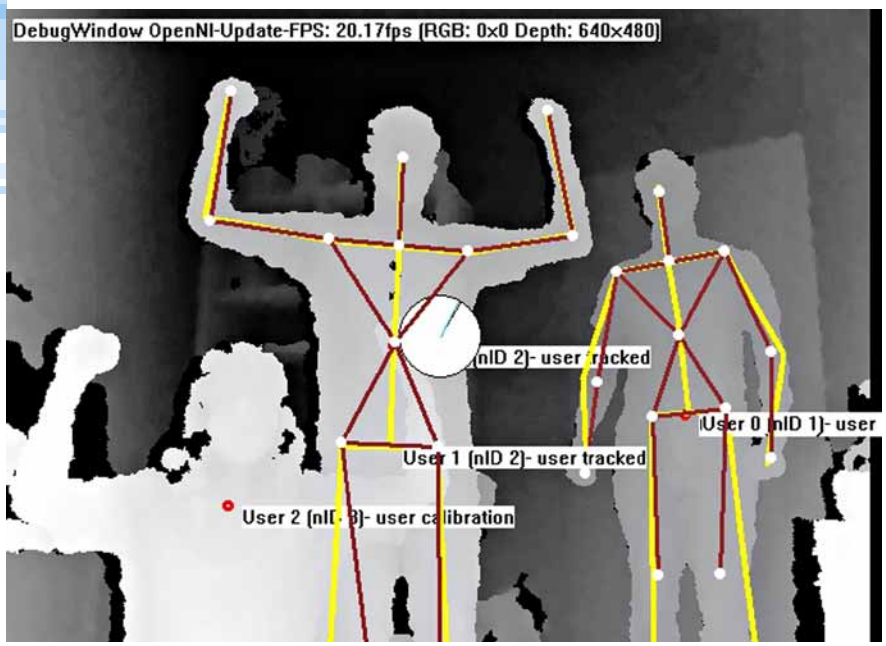


Abbildung 7: Die Debugging-Ausgabe mit dem Tiefenbild der Kinect und den getrackten Spielern. Außerdem sind die erfassten Gelenkpunkte (weiß) mit den Rohdaten (braun) und den aufbereiteten Skelettdaten für das Spiel (gelb) zu erkennen.

Beine des Spielers nicht erfasst werden, muss deren Lage auf andere Weise ermittelt oder geschätzt werden. Somit gibt es Gelenkpunkte, die von der Trackingsoftware zuverlässiger erfasst werden als andere. Dieses mehrfache »Filtern« der Bewegungsdaten führte zu immer komplexeren Algorithmen und größeren Abhängigkeiten der Knochendaten untereinander. So wurde es immer schwieriger, Fehler in den aus Matrizen bestehenden Datenströmen zu finden. Und uns wurde bewusst, dass wir uns nicht auf Konsolenausgaben und den Debugger beschränken konnten. Wir entwickelten daher eine Debugging-Ausgabe, mit der alle nötigen Informationen über die Szene grafisch dargestellt werden konnten (**Abbildung 7**). Damit waren wir in der Lage, auf einen Blick zu überprüfen, welche Spieler gerade erfasst wurden, in welchen Zuständen sie sich befanden und welche Spieler-IDs sie erhielten. Außerdem konnten Roh- und Filterdaten verglichen und die Framerate des Spiels mit der des Kamera-Frameworks in Beziehung gesetzt werden. Die visuelle Darstellung unterstützte im Endeffekt die Auswertung großer Zahlenmengen.

Wir machten außerdem von der Möglichkeit Gebrauch, Videoströme aufzuzeichnen und wieder abzuspielen. Wenn ein Fehler in unserer Debugging-Ausgabe auftauchte, musste die aufgezeichnete Sequenz immer wieder abgespielt werden, bis die Ursache eingegrenzt werden konnte. Wir konnten zusätzlich durch die aufgezeichneten Sequenzen springen. Allerdings gab es auch Fehler, die nur aus einem bestimmten Kontext heraus entstanden. Um diese zu reproduzieren, musste dann doch die gesamte Sequenz immer wieder gesichtet werden. Dieses Verfahren hat eine weitere Schwäche: Wenn etwa bestimmte Situationen zu Abstürzen führen, können diese mit der Methode in der Regel nicht reproduziert werden, weil der Absturz die Aufzeichnung an der entscheidenden Stelle unbrauchbar macht.

Multiplayer-Handling

Der Mehrspielermodus fand erst später seinen Weg ins Spiel. Wir ließen das Feature zu Beginn völlig außer Acht, denn das MoCap-System der Hochschule konnte ohnehin nur eine Person erfassen. Für eine intuitive Mehrspielerbehandlung waren somit zeitaufwändige Umstrukturierungsarbeiten notwendig.

Die eigentliche Problematik ist erkennbar, wenn man sich die Unterschiede der Steuerungssysteme vor Augen führt. Im Gegensatz zu einer klassischen Steuerung, bei der jeder Spieler ein Eingabegerät in den Händen hält und sich als Spieler 1, 2, 3 oder 4 registriert, müssen bei der Kinect alle Personen von einer Kamera unterschieden und getrackt werden (**Abbildung 8**). Weil wir auf die mitgelieferte Trackingsoftware der Kinect-Frameworks zurückgriffen und diese nicht selbst entwickelt hatten, mussten wir uns mit deren Eigenheiten arrangieren. So besitzt die Trackingsoftware die Fähigkeit, Personen zu erkennen, sobald sie sich in den Sichtbereich der Kamera bewegen. Dieses Prinzip führt jedoch zu erheblichen Problemen im Mehrspieler- als auch im Einzelspielermodus. Um einen Spieler klar von seiner Umgebung zu trennen und ein Startereignis zu generieren, ist es daher sinnvoll, Startposen zu verwenden. Damit kann ausgeschlossen werden, dass beispielsweise eine sich bewegende Tür als Person erfasst wird. Des Weiteren kann es zu Verwechslungen der Spieler kommen. Denn es ist schwierig für die Trackingsoftware, Personen auseinanderzuhalten, die eng beieinanderstehen. Dies sind typische Fehler von Trackingprogrammen bei der Interpretation der gefilmten Szene, die auch in Zukunft wahrscheinlich nie komplett ausgeschlossen werden können.

Wie werden die Spieler den Spielfiguren zugewiesen? Diese auf den ersten Blick trivial erscheinende Frage bezüglich des Multiplayer-Handlings warf bei der Umsetzung noch weitere Fragen auf. Wird derjenige, der zuerst ins Bild tritt, zum Spieler 1 und die Person danach zum Spieler 2 oder sortiert man die Spieler von links nach rechts? Was passiert, wenn ein Spieler den Trackingbereich verlässt? Rücken die anderen Spieler automatisch nach oder wartet die Spielfigur auf einen neuen Spieler? Was passiert, wenn er wieder den Trackingbereich betritt? Es gibt noch unzählige solcher Fragen, die in Abhängigkeit vom Kontext auch unterschiedlich beantwortet werden können. Im Rahmen von SkiRanger konnten wir die für uns relevanten Fallbeispiele mittels durchdachter Konzepte sowie Trial & Error lösen.

Die fehlende Vertriebsplattform

Wie bereits erwähnt war es unser Ziel, SkiRanger zu einem vollwertigen Produkt zu entwickeln,

das auch vertrieben werden konnte. Daher beobachteten wir während der Kinect-Portierung den sich stetig ändernden Kinect-Markt und hielten nach möglichen Partnern Ausschau. Dabei mussten wir feststellen, dass sich der ASUS Xtion Store noch im Aufbau befand und Vertriebsplattformen wie Steam oder Desura noch keine Möglichkeit boten, PC-Kinect-Spiele zu platzieren. Auch einer Veröffentlichung von SkiRanger auf der Xbox 360 versperren einige lizenzrechtliche und firmenpolitische Hindernisse den Weg, die wir aus eigener Kraft nicht überwinden können. Zum Beispiel gibt es von Seiten Microsofts keine Kinect-Unterstützung für die Indie-Plattform der Xbox 360, auf der man kostengünstig sein Spiel veröffentlichen könnte. Xbox Live Arcade besitzt Kinect-Unterstützung, bleibt aber größeren Entwicklerstudios vorbehalten. Hinzu kommen die relativ hohen Lizenzgebühren für die Nutzung der Unreal Engine 3 auf der Xbox 360. Als Indie befindet man sich also in einem Dilemma. Um SkiRanger auf der Xbox 360 zu veröffentlichen, bräuchte es in jedem Fall einen finanzkräftigen Investor. Aus diesen Gründen setzten wir vorerst unsere Hoffnungen auf den PC-Kinect-Markt und wollten es unabhängig von den genannten Vertriebsplattformen auf eigene Faust versuchen.

Die Betaversion von SkiRanger sollte Free2-Play sein, um die Hemmschwelle für neue User möglichst niedrig zu halten. Bei erfolgreicher Betaphase würden wir zu einer kostenpflichtigen Version wechseln, die durch immer neue Updates an Umfang gewinnen würde. Deshalb entwickelten wir ein eigenes System, um unsere User kontinuierlich mit Updates versorgen zu können und ihnen zu zeigen, dass unser Projekt voranschreitet. Leider entwickelte sich der Kinect-Markt nicht zu unseren Gunsten. Obwohl SkiRanger einer der wenigen PC-Kinect-Titel ist und für uns praktisch keine Konkurrenz existiert, kann das Spiel mangels Vertriebsplattform auf dem PC nicht erfolgreich sein und wird wohl ein Freizeitprojekt bleiben.

Bei den Sensorherstellern scheint ebenfalls nur wenig Interesse zu bestehen, die Voraussetzungen für eine PC-Kinect-Plattform zu schaffen. So widmet Microsoft seiner Kinect for Windows

Abbildung 8: Der Multiplayer-Modus von SkiRanger mit eingeblendetem Mitspieler-Tracking.

nur wenig Aufmerksamkeit, und der ASUS Xtion Store hat es nicht über seine 20 Releasetitel geschafft. Die erwähnten fehlenden Standards tragen ihren Teil dazu bei. Mittlerweile versuchen auch andere Portale wie Mageca.com, Kinect-Anwendungen eine Vertriebsplattform zu bieten, doch bis jetzt leider nur mit mäßigem Erfolg.

Das Ergebnis: SkiRanger Beta 1.4.7

Die Kinect-Steuerung ist ein noch unentdecktes Land, es können noch viele Spielkonzepte entwickelt werden. Wir haben versucht, Klassisches und Neues mit der Bewegungssteuerung zu kombinieren. So kann man in SkiRanger beispielsweise in »Tony Hawk«-Manier Pipes und Grinds unsicher machen. Wir arbeiteten außerdem an einer Art »Adventure Bowling«-Modus, in dem der Spieler mit großen Schneebällen die »bösen« Schneemänner wegputzen muss. Dieser Spielmodus hat es bis jetzt leider noch nicht ins Spiel geschafft (**Abbildung 9**).

Auch wenn es nur die für uns wichtigsten Features ins Spiel geschafft haben, ist SkiRanger ein vorzeigbares Partyspiel geworden, inklusive Stunts, Turbos und Schneemännern. Bis zu vier Spieler gleichzeitig können in drei Modi gegeneinander antreten. Außerdem bietet SkiRanger ein weitaus intensiveres Skifahrerlebnis als so manches offizielle Skispiel für Kinect – wenn man den Nutzern glauben darf.

Henrik Weirauch, Toni Seifert

Abbildung 9: Für SkiRanger haben wir versucht, etablierte Mechanismen wie das Grinden aus Skate- und Snowboard-Spielen mit Bewegungssteuerung umzusetzen (links). Auf dem rechten Bild sieht man den Prototyp für unseren Schneeballmodus, in dem man böse Schneemänner umkegeln muss.